|  |  |
| --- | --- |
| **Min-Max in C++** | |
| #include <iostream>  #include <climits> // for INT\_MAX and INT\_MIN  using namespace std;  int getMin(int arr[], int i, int n) {      if (n == 1) {          return arr[i];      } else {          return min(arr[i], getMin(arr, i + 1, n - 1));      }  }  int getMax(int arr[], int i, int n) {      if (n == 1) {          return arr[i];      } else {          return max(arr[i], getMax(arr, i + 1, n - 1));      }  }  int main() {      int arr[] = {12, 8, 45, 67, 9};      int n = sizeof(arr) / sizeof(arr[0]);      cout << "Minimum element of array: " << getMin(arr, 0, n) << endl;      cout << "Maximum element of array: " << getMax(arr, 0, n) << endl;      return 0;  } | 🧾 Dry Run Table for getMin(arr, 0, 5)  | **Call Level** | **i** | **arr[i]** | **Recursive Call** | **Returned Value** | **Computation** | | --- | --- | --- | --- | --- | --- | | 1 | 0 | 12 | min(12, getMin(1, 4)) | 8 | min(12, 8) | | 2 | 1 | 8 | min(8, getMin(2, 3)) | 8 | min(8, 9) | | 3 | 2 | 45 | min(45, getMin(3, 2)) | 9 | min(45, 9) | | 4 | 3 | 67 | min(67, getMin(4, 1)) | 9 | min(67, 9) | | 5 (base) | 4 | 9 | return arr[4] | 9 | Base case |  🧾 Dry Run Table for getMax(arr, 0, 5)  | **Call Level** | **i** | **arr[i]** | **Recursive Call** | **Returned Value** | **Computation** | | --- | --- | --- | --- | --- | --- | | 1 | 0 | 12 | max(12, getMax(1, 4)) | 67 | max(12, 67) | | 2 | 1 | 8 | max(8, getMax(2, 3)) | 67 | max(8, 67) | | 3 | 2 | 45 | max(45, getMax(3, 2)) | 67 | max(45, 67) | | 4 | 3 | 67 | max(67, getMax(4, 1)) | 67 | max(67, 9) | | 5 (base) | 4 | 9 | return arr[4] | 9 | Base case |  ✅ Final Output: Minimum element of array: 8  Maximum element of array: 67 |
| Output:- Minimum element of array: 8  Maximum element of array: 67 | |